Empir Software Eng (2015) 20:82–109
DOI 10.1007/s10664-013-9277-5

HAZOP-based identification of events in use cases

An empirical study

Jakub Jurkiewicz · Jerzy Nawrocki · Mirosław Ochodek · Tomasz Głowacki

Published online: 27 September 2013

Abstract Completeness is one of the main quality attributes of requirements specifications. If functional requirements are expressed as use cases, one can be interested in event completeness. A use case is event complete if it contains description of all the events that can happen when executing the use case. Missing events in any use case can lead to higher project costs. Thus, the question arises of what is a good method of identification of events in use cases and what accuracy and review speed one can expect from it. The goal of this study was to check if (1) HAZOP-based event identification is more effective than *ad hoc* review and (2) what is the review speed of these two approaches. Two controlled experiments were conducted in order to evaluate *ad hoc* approach and H4U method to event identification. The first experiment included 18 students, while the second experiment was conducted with the help of 82 professionals. In both cases, accuracy and review speed of the investigated methods were measured and analyzed. Moreover, the usage of HAZOP keywords was analyzed. In both experiments, a benchmark specification based on use cases was used. The first experiment with students showed that a HAZOP-based

Communicated by: John C. Knight

This project was funded by Polish National Science Center based on the decision DEC-2011/01/N/ST6/06794 and by the PUT internal grant DS 91-518.

J. Jurkiewicz (✉) · J. Nawrocki · M. Ochodek · T. Głowacki
Poznan University of Technology, Institute of Computing Science,
ul. Piotrowo 2, 60-965 Poznań, Poland
e-mail: jakub.jurkiewicz@cs.put.poznan.pl

J. Nawrocki
e-mail: jnawrocki@cs.put.poznan.pl

M. Ochodek
e-mail: mochodek@cs.put.poznan.pl

T. Głowacki
e-mail: tglowacki@cs.put.poznan.pl

review is more effective in event identification than *ad hoc* review and this result is statistically significant. However, the reviewing speed of HAZOP-based reviews is lower. The second experiment with professionals confirmed these results. These experiments showed also that event completeness is hard to achieve. It on average ranged from 0.15 to 0.26. HAZOP-based identification of events in use cases is an useful alternative to *ad hoc* reviews. It can achieve higher event completeness at the cost of an increase in effort.

Keywords Requirements engineering · Software quality · Use cases · HAZOP · Controlled experiment

1 Introduction

Use cases introduced by Jacobson (1985) have gained a lot of attention in the software engineering community since their introduction in the 1980's (Neill and Laplante 2003). Roughly speaking, a use case is a sequence of steps augmented by a list of events and the alternative steps associated with them that respond to those events. Both the steps and the events are written in a natural language. This makes reading and understanding software requirements specification easier, especially if the readers are IT laymen.

One of the main attributes that can be assigned to use cases (and any other form of requirements) is completeness (IEEE 1998). In the context of use cases, completeness can have several meanings: e.g. functional completeness (the set of use cases covers all the functional features of the system) or formal completeness (there are no TBDs— IEEE 1998). In this paper the main concern is event-completeness: requirements specification presented in the form of use cases is *event-complete* if all the events that can occur during the execution of the use cases are named in this specification.

Stark et al. (1999) indicate that missing error handling conditions account for most software project rework. This increases project costs and also leads to delayed delivery. In the context of use cases, these *error handling conditions* relate directly to use-case events. Moreover, many authors (Mar 1994; Wiegers 2003a; Carson 1998; Cox et al. 2004; Adolph et al. 2002a) indicate that the list of events should be complete, e.g. Adolph et al. (2002a) propose a pattern called *Exhaustive Alternatives* which states *Capture all failure and alternatives that must be handled in the use case*. Thus, one needs an effective method of checking the event-completeness of a given set of use cases. Such a method would also be useful in the context of the spiral requirements elicitation recommended by Adolph et al. (2002a). Their approach is based on the *breadth before depth* pattern. Firstly, they suggest listing all the actors and their goals, then adding a main success scenario to each goal. After a pause, used to reflect on the requirements described thus far, one should continue identifying the events and adding alternative steps for each event. In this paper, two methods of events identification are studied experimentally: *ad hoc* and H4U (HAZOP for Use Cases). The former does not assume any particular process or tools: The Analyst is given a set of use case scenarios and he/she has to identify the possible events in any way he/she preferes. The second method is based on the HAZOP review process (Redmill et al. 1999).

The paper aims at answering the following questions: (1) What level of completeness can one expect when performing events identification (with H4U and *ad hoc* approaches)? (2) How should we perform events identification in order to obtain the maximum completeness? (3) How can we minimize the effort necessary to obtain the maximum level of completeness?

The paper starts with a short presentation of the 9-step procedure to elicit requirements specification in Section 2. That procedure is an implementation of the *breadth before depth* strategy recommended by Adolph et al., and it shows a context for events identification (however, other methods of requirements specification based on use cases can also benefit from H4U). Details of the H4U method along with the example are presented in Section 3. To evaluate H4U against *ad hoc* review, two controlled experiments have been conducted: with students and with practitioners. The design of the experiments and their results are presented in Section 4. Both experiments focus on individual reviews. An interesting aspect of the conducted experiments is usage of HAZOP keywords. Those data are presented in Section 5. Related work is discussed in Section 6.

2 An Evolutionary Approach to Writing Use Cases

2.1 Overview of Use Cases

A use case consists of five main elements (see Fig. 1): (1) a header including the name of a use case; the name indicates the goal of the interaction; (2) a main scenario describing the interaction between the actor and the described system; the main scenario should present the most typical sequence of activities leading to achievement of the goal of the interaction; (3) extensions consisting of (4) events that can occur during the execution of the main scenario and (5) alternative steps taken in response to the events. The focus of this paper is on the fourth element of use cases, i.e., the events.

Moreover, use cases can specify requirements on different levels: summary goals, user goals, subfunctions (Cockburn 2000). The user goals level is the most important one as it shows the main interactions between the user and the system. Events in use cases at this level should relate either to decisions made by the user regarding the path taken to obtain his/her goal or obstacles which the user can encounter when performing actions leading him/her to achieving his/her goal. Use cases present functional behaviour requirements and should not include information related to architectural decisions or non-functional requirements (NFRs), such as events related to the performance or to internal architectural components. Non-functional requirements are specified separately using, for instance, ISO 25010 characteristics (ISO 2011).

As with every other element, the form and the quality of events depend heavily on the skills of the author of a use case. However, some experts (Adolph et al. 2002a) have proposed best practices concerning writing high quality use-case events. The most important practice is called *Detectable Conditions*. It suggests describing only those events which can be detected by the system. If the system cannot detect an event, then it cannot react to it; hence, specifying those events is useless as they can blur the use case.

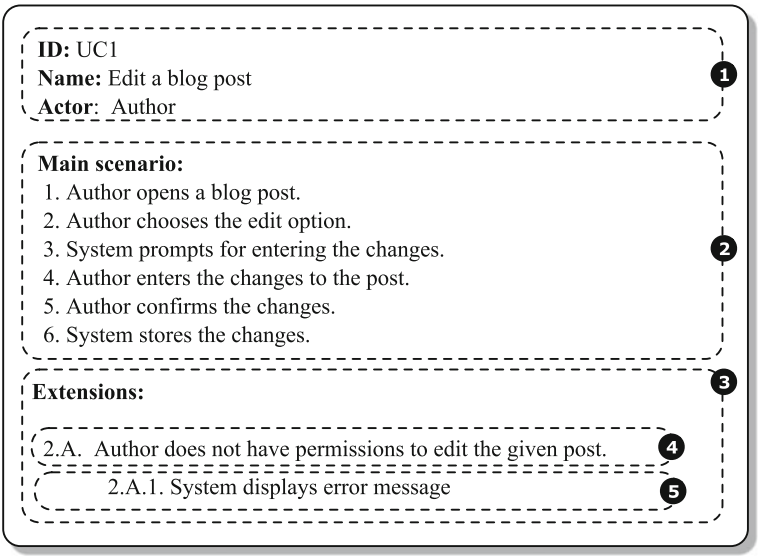

Fig. 1 An exemplary use case with a header (*1*), main scenario (*2*), extensions (*3*) consisting of an event (*4*) and an alternative sequence of steps (*5*)

2.2 Process of Use-Case Elicitation

There are two contexts in which event identification is useful: review of use cases and elicitation of use cases.

In the context of use-case review, a reviewer (or an inspection team) is given a set of use cases and he/she tries to determine if those use cases are event-complete. In order to do this, one could identify a set of events associated with each step and check if every event from this set has its counterpart in the specification under review.

In the context of use case elicitation, the practice of *spiral development* suggested by Adolph et al. (2002b) can be very useful. It can be supported by another practice called *breadth before depth*: first an overview of use cases should be developed (almost all the use cases should be named), and detailed description should be added at the next stage. These two practices go well with the approach proposed by Cockburn (2000): *"Expand your use cases into Scenario Plus Fragments by describing the main success scenario of each use case, naming the various conditions that can occur, and then fleshing out some of the more important ones"*.

In our work the patterns of *spiral development* and *breadth before depth* have been elaborated into a 9-step process for requirements specification elicitation. That process supports the XPrince methodology (Nawrocki et al. 2006) and is presented in Fig. 2 as a use case. Description of the problem, which is the main outcome of Step 1, is based on RUP's project statement (Kruchten 2003). The business process described in Step 2 can be extended if necessary with other forms of knowledge description, such as examples of information objects (e.g. Invoice) if the domain requires this level of detail. A context diagram created in Step 3 allows better understanding of the overview of the system, the actors interacting with the system, and the interfaces which the system will need to use. The use-case diagrams developed

Name: Elicite functional use cases
Actor: Analyst
Main scenario:
1. Analyst describes the problem, its implications and the proposed solution.
2. Analyst acquires domain knowledge and presents it as a set of business processes.
3. Analyst outlines the solution with a context diagram.
4. Analyst extends the context diagram to a use-case diagram.
5. Analyst identifies domain objects through analysis of domain knowledge described in step 2.
6. Analyst checks the completeness of the use cases operating the domain objects.
7. Analyst provides a main scenario for each use case.
8. Analyst annotates each use-case step with events that may possibly occur when the step is executed.
9. Analyst provides an alternative sequence of steps for each event.

Fig. 2 The 9-step process of use-case elicitation presented as a use case

in Step 4 present the goals of the actors. The domain objects, identified in Step 5, are the objects the system will manipulate with or present to the user. These objects represent the physical objects of the real world. Having those objects the checking of completeness of the set of use cases from the point of view of CRUD (Create, Retrieve, Update, Delete) (Martin 1983). In Step 6 each use case manipulating a domain object is classified as C, R, U or D, and if any of those letters is missing for a given domain object, then the analyst should provide an explanation or extend the set of use cases to include the missing operation (this kind of analysis was proposed by Wiegers (2003b)). In Step 7 a main scenario for each use case is provided. Events which may occur when main scenarios are executed are identified in Step 8; here the proposed method (H4U) is used. Step 9 completes each use case with alternative steps assigned to each event.

From the above it follows that whenever the analyst wants to review use cases or elicit them in a systematic way he or she will be confronted with the problem of events identification.

2.3 The Problem

In this paper the focus is on step #8, identification of events which may occur when the main scenario is executed. To be more precise, one can formulate the problem in the following way:

Problem of events identification: *Given a step of a use case, identify a set of events that may occur when the step is executed. Identify all possible events assuming that the system is formally correct and exclude the events related to non-functional aspects of the system.*[1]

[1] In order to facilitate the identification of events in the experiments presented in Section 4, we gave the participants the freedom to identify any kinds of events they wanted to. However, it resulted in a visible number of events that had to be rejected from the analysis.

Obviously, the events unrelated to the step, but occurring in parallel to it just by chance, should be excluded—we are only interested in the cause-effect relationship between an event and a step in the main scenario that causes the change in the executed scenario.

There can be several approaches to the problem of events identification. Typically events in use cases are identified as a result of an *ad hoc* process by an analyst or customer who try to guess what kind of events or exceptions can occur while use-case scenarios are being performed. However, a special approach, oriented strictly towards the problem of events identification could also be used. An example of such a method is H4U (discussed in details in Section 3). Based on this method the following goal arises: *Analyze the methods of event identification for the purpose of evaluation with respect to accuracy and speed from the point of view of an analyst.*

3 HAZOP Based Use-Case Review

The proposed H4U method for events identification is based on a well-known review approach for mission critical systems called HAZOP (Redmill et al. 1999) (short for H4U which stands for *HAZOP for Use Cases*). The sections below describe the details of the H4U method and discuss how it differs from the original HAZOP approach.

3.1 HAZOP in a Nutshell

HAZOP (Hazard and Operability Study) is a structured technique for the identification and analysis of hazards in existing or planned processes and designs (Lawley 1974; CIA 1992; Leveson 1995; Schlechter 1995; Redmill et al. 1999). Moreover, HAZOP allows for the distinguishing of possible causes of hazards and their implications, together with available or future protection steps. The method was initially intended for the analysis of chemical plants. Later it was adapted for analysis of computer-based systems.

The method involves a team of experts who use special keywords that guide them through identification of so called deviations (deviations are counterparts of events in use-case steps). There are two kinds of keywords: primary and secondary. Primary keywords refer to attributes (Temperature, Pressure, etc.). Secondary keywords indicate deviations (NO, MORE, etc.). Considering a fragment of a chemical or IT system, the inspection team of experts tries to interpret a pair of keywords (e.g., Pressure - MORE could be interpreted as too much pressure in a pipe).

HAZOP uses the following secondary keywords (Redmill et al. 1999):

- NO: the design intend is not achieved at all,
- MORE: an attribute or characteristic exceeds the design intend,
- LESS: an attribute or characteristic is below the design intend,
- AS WELL AS: the correct design intent occurs, but there are additional elements present,
- PART OF: part of the design intend is obtained,
- REVERSE: the opposite of the design intent occurs,

Item	Attribute	Keyword	Cause	Consequences	Protection	Questions

Fig. 3 HAZOP report sheet

- OTHER THAN: there are some elements present, but they do not fulfill the design intend,
- EARLY: the design intent happens earlier than expected,
- LATE: the design intent happens later than expected,
- BEFORE: the design intent happens earlier in a sequence than expected,
- AFTER: the design intent happens later in a sequence than expected.

At the end of a HAZOP session, a report is created (see Fig. 3). The report includes, among other things, two columns: Attribute (primary keyword) and Keyword (secondary keyword), which together describe an event that can occur when the system is running.

3.2 H4U: HAZOP for Use Cases

To solve the problem of events identification, adapting the HAZOP method is proposed. The first question is how to interpret the terms "design intent" and "deviation". The H4U method is targeted at analysis of use cases, therefore:

- *design intent* is the main scenario of a use case;
- *deviation* is an event that can appear when the scenario is executed and results in an alternative sequence of steps.

There two kinds of deviations: functional and non-functional. Functional deviations from the main scenario are all those events that can appear even when the system works correctly. An example could be a trial to buy a book which is not in the store or paying with an expired credit card. Non-fuctional deviations follow from a violation of nonfunctional requirements such as correctness, availability, security, etc. (see e.g. ISO 25010 characteristics ISO 2011). Examples include user session timeout, unacceptable slowdown, etc. Those events shouldn't be specified within use cases because they are cross-cutting and could appear in almost all the steps, which would make a use case *unreadable*. Therefore, the deviations as well as the design intent should be kept separate as they belong to two different dimensions. Nonfunctional deviations could also be identified using HAZOP but, according to the separation of concerns principle, they should be specified elsewhere and therefore they are out of the scope of the H4U method.

The previously presented HAZOP secondary keywords are generic keywords used in process industries with the addition of the last four keywords introduced in Redmill et al. (1999) where HAZOP for software is presented. These four keywords were introduced in order to analyze the timing aspect of software. It was decided not to alter the presented set of secondary keywords. There are other sets of keywords that could be used, e.g. the guide words from the SHARD method proposed by

ID: UC1
Name: Identification of an event
Actors: Reviewer (can also be a Review Team)
Main flow:
1. Facilitator presents the H4U method to the Reviewer.
2. Facilitator provides the Reviewer with a set of main scenarios of use cases.
3. Reviewer reads the main scenarios step by step.
4. Reviewer uses primary and secondary to identify the events and annotates the steps with the events.
5. Reviewer presents the results the Facilitator.

Fig. 4 Overall process of the H4U method

Pumfrey (1999), however, the aim of this paper is to investigate how the well known HAZOP method can influence events identification.

The original HAZOP primary keywords are specific to chemical installations and they should be replaced with a set which would fit computer-based systems. To achieve this, a set of 50 industrial use-case events were analyzed by two experts. As a result the following primary keywords have been identified:

- Input & output data—relates to objects that are exchanged between external actors and the system;
- System data—relates to objects already stored in the system;
- Time—relates to the "moment of time" when a step is being executed (e.g., dependencies between actions, pre- and post- conditions of actions); it does not relate to non-functional aspects such as performance (e.g. response time) or security (user session timeout).

To check the completeness of the above set of primary keywords, a small empirical study has been performed. 300 use-case events have been investigated by experts in the field, and it was possible to present each of them as a pair of one of the proposed primary keywords and standard secondary keywords; thus, one can assume that the proposed set of primary keywords is complete. Since the set of primary keywords consists of only 3 elements, it is easy to memorize them; therefore there is no need to include them as a checklist during the review process.

The H4U method can be used in two ways: as an individual review method and as a team review method. The overall process of the H4U method is presented in the form of a use case in Fig. 4. As an output, a report is produced according to the H4U report table presented in Fig. 5. The report should consist of the following

Use case ID	Step number	Primary keyword	Secondary keyword	Event

Fig. 5 H4U report table

Use Case ID	Step number	Primary Keyword	Secondary Keyword	Event
UC1	4	Input & output data	NO	*Author did not provide data.*
UC1	4	Input & output data	MORE	*Author exceeds the limit of characters per single post.*
UC1	4	Time	LATE	*Author changes the post after the allowed time passed.*

Fig. 6 Exemplary results of applying the H4U method to one use-case step

elements: *Use case ID*—ID of a use case, *Step number*—the number of the step in the main scenario, *Primary keyword*—the primary keyword used to identify an event, *Secondary keyword*—the secondary keyword used to identify an event, *Event*—the identified event.

3.3 Example

This section presents an example of how the H4U method could be used to identify events in an exemplary use case from Fig. 1. An analyst goes through every step of the use case and uses the proposed primary and secondary keywords in order to identify as many events as possible. Let us assume that the analyst is currently analyzing Step 4: *Author enters the changes to the post*. Events which might be identified using the primary and secondary keywords analysis are presented in Fig. 6.

4 Experimental Evaluation of Individual Reviews

In order to evaluate the process of events identification in use cases, we decided to conduct a series of controlled experiments.

The goal of the experiments was to analyze H4U and *ad hoc* use-case events identification; for the purpose of evaluation; with respect to their review speed and accuracy; from the point of view of both students and experienced IT professionals; in the context of a controlled experiment performed at university and in industry.

The first experiment (Experiment 1) was conducted in academic settings to investigate how the structured and *ad hoc* approaches to events identification support the work of students of software engineering.

The second experiment (Experiment 2) was conducted with the participation of professionals, in order to observe how the methods affect the work of experienced reviewers who work professionally on software development.

Both experiments were conducted according to the same design; therefore, we will present the details only in the description of Experiment 1, and discuss the differences in the description of Experiment 2. The results of both experiments were independently analyzed by two researchers. Every disagreement was discussed until a decision about the problematic element was made.

4.1 Empirical Approximation of Maximal Set of Events

In order to be able to analyze and interpret the results of the experiments a maximal set of all possible events in the software requirements specification being an object of the experiments has to be agreed. After removing irrelevant events, such a set is to be used to assess the accuracy of events identification process. The procedure of removing irrelevant events will be discussed in the following sections.

Three approaches to defining the reference set of all possible (acceptable and irrelevant) events were considered:

- A1: include events that were defined by the authors of benchmark specification;
- A2: the authors of the paper could brainstorm in order to identify their own reference set of possible events;
- A3: construct the approximate maximal set in an empirical way, based on all distinct events identified by the participants in the experiments (after reviewing them).

We decided to choose approach A3, because we assumed that a group containing around 100 people would be able to generate the most exhaustive set of events. If we had decided to accept the approaches A1 or A2, we would had the potential problem of interpreting the proposals of events provided by the participants that were reasonable (events that could possibly appear in this kind of system) but neither the authors of the paper nor the authors of the use-case benchmark could identify them. After the experiment, it turned out that it was a good choice, because the set of events defined by the participants of the experiments contained all the events that were originally present in the benchmark use cases.

4.1.1 Rejecting Undetectable Events

As it was mentioned in Section 2, only the events which may be detected by a system are considered valid. For example the event *"system displayed a wrong form"* is apparent to the user, but the system cannot detect it and react to it. These kinds of events were very often related to possible defects (bugs) in the system. All the participants (in both experiments) proposed in total 4,024 events during the experiments. Based on the initial analysis, 1,593 events were rejected as being undetectable or invalid, therefore 2,431 events were taken into consideration in further analysis.

4.1.2 Handling Duplicated Events

After the experiments, we analyzed every event in the context of the use case it belongs to and the use-case step by which it is triggered. As a result an abstract class was assigned to such an event; the class represents its semantic meaning (the same abstract event can be expressed in different ways using natural language). It was noticed that some participants had included a number of abstract classes of events in one event description. Moreover, during the analysis it appeared that the participants had employed different approaches to assign events to use-case steps. Let us consider an exemplary part of the use-case scenario presented in Fig. 7.

An exemplary list of events that could be identified by participants, together with the identifiers of steps the events were assigned to, are presented in Fig. 8. It can be

4. …
5. Candidate fills and submits the form.
6. System verifies if data is correct.
7. System informs Candidate that account has been created.
8. …

Fig. 7 Example of groups of steps in a use case

noticed that all the proposed events describe the problem of *missing data*, and all of them could be assigned to each of the three steps. Therefore, it does not really matter to which of the three steps the event is assigned to, however, all of them should be treated as the same event. We have to mention that some of the participants in the similar cases assigned the same type of event to all of the steps, i.e. they wrote down three separate events. Unfortunately, this could visibly influence the results of the experiments, as participants who had written down the same events many times for closely related steps would have a higher number of identified events, even though the events carried the same meaning.

To solve this problem we decided to identify groups of consecutive steps that are indistinguishable in terms of events that can be assigned to them. It was decided to treat this kind of group of steps as a whole: if participant assigned an event to any of the steps from the group it was treated as if the event was assigned to the whole group; if the participant assigned an event to more than one step from the group it was considered as a single assignment to the whole group. The requirements specification, being an object of the experiments, consisted of 33 use cases and 113 groups of steps (and 156 individual steps).

After this phase of analysis, 2510 abstract classes of events assigned to events descriptions were identified.

4.1.3 Excluding Non-Use-Case Events

As a result of the analysis of the abstract classes of events assigned to groups of steps, we observed that some classes of valid events (which could be detected by the system) should not be included in descriptions of use cases on user goals level or should be excluded from the analysis.

Most of such events related to non-functional requirements, which should be specified separately from use cases (see the explanation in Section 3.2).

The second group of events that we decided to exclude from the analysis extended main scenarios with some auxiliary functions. The participants in the experiments

Event	Step number
Candidate doesn't provides all necessary data	5
User didn't provide all required data	6
Candidate provided no data	7

Fig. 8 Example of events that could be identified for a group of use-case steps presented in Fig. 7

could use their imagination to propose any number of such extensions, which would make it extremely difficult to decide whether or not we should include them in the analysis.

The rejected classes of events are presented in Table 1, together with examples and the justification for excluding them.

4.1.4 Events Included into the Maximal Set of Events

In the end 1,550 abstract classes of events assigned to events descriptions were left for construction of the approximation of the maximal set of events. The list of accepted events classes is presented in Table 2.

4.2 Experiment 1: Student Reviewers

4.2.1 Experiment Design

The *independent variable* considered in the experiment was a method used to identify events in use cases (two values were considered: *ad hoc* and H4U). Two *dependent variables* were defined: the speed of the review process and its accuracy. The speed was measured as the number of steps in the use cases that were analyzed, divided by the time required to perform the analysis (see (1)).

$$Speed(p) = \frac{\#steps(p)}{T}[steps/minute] \tag{1}$$

Where:

- p is a participant in an experiment;
- $\#steps(p)$ is the number of steps reviewed by the participant p;
- T is the total time spent on the review (constant: 60 min).

The accuracy was measured as a quotient between the number of events identified by a participant compared with the total number of distinct events identified by the participants of both experiments in the steps reviewed by a given participant (see (2)).

$$Accuracy(p) = \frac{\sum_{s=1}^{distance(p)} \#events(p,s)}{\sum_{s=1}^{distance(p)} \#Events(s)} \tag{2}$$

Where:

- p is a participant in an experiment;
- $distance(p)$ is the furthest step in the specification that was analyzed by the participant p;
- $\#events(p,s)$ is the number of distinct events identified by the participant p based on the step s;
- $\#Events(s)$ is the number of distinct events identified by all participants in both experiments based on the step s.

Participants The participants in the experiment were 18 4th and 5th-year students of the Software Engineering programme at Poznan University of Technology. The

Table 1 Abstract classes of events that should not be included into use-case descriptions (they were excluded from the analysis)

Abstract class of event	Explanation	Example	Number
Authorization problem	Events concerning authentication and authorization problems. They should not be included in alternative flows of a user-level use case, but handled separately as NFRs.	*User does not have permission to add* a new thread.	42
Concurrency of operations	Events related to concurrency problems at the level of internal system operations (e.g., concurrent modification of the same object). This kind of requirement relates to NFRs (transactional operations). There are also concurrent operations that relate to business rules (e.g., while one is applying to a major the admission is closed). This kind of events belong to another abstract class of events and was accepted.	*Candidate data was changed by the administrator before the candidate submitted modif ications.*	41
Connection problem	Events describing problems with the connection between a user and the system. Use-cases should not include technical details. It should be described as an NFR. The system is not able to respond to such an event.	*The data could not be sent to the server.*	57
Internal system problem	Events describing failures inside the system. These events should not be included as user-level use cases.	*System was not able to store the data in the database.*	339
Linked data	Problems related to the consistency of data stored in the system. It should rather be described in the data model, business rules, or NFRs.	*Administrator selects item to be removed that is connected to other items in the system.*	7
Other action	Events expressing actor's will to perform a different action than the action described in a use-case step. Any participant could come up with an enormous number of such events. We would not be able to assess if they are reasonable and required by the customer.	*User wants to view the thread instead of adding a new one.*	226
Timeout	Events referring to exceeding the time given to complete a request (e.g., user session timeout). Should be describe as an NFR.	*It takes candidate too long to fill the form.*	68
Withdraw	Events describing cases when a user wants to resign from completing a running operation. This events could be assigned to almost every step of a use case, so this would make the use case cluttered.	*User does not want to provide the data, so he/she quits.*	180

Table 2 Abstract classes of events that should be included in use-case descriptions (they were included in the analysis)

Abstract class of event	Explanation	Example	Number
After deadline	The operation is not allowed, because it is performed after deadline.	*The admission for the chosen major is closed.*	12
Algorithm runtime error	Problems with the execution of an admission algorithm, which is defined by a user.	*Unable to execute the algorithm.*	9
Alternative response	Events describing the alternative response of an actor (usually, a negative version of the step in the main scenario)	*The committee reject the application.*	89
Connection between actors	Problems with communication between the system and external actors, i.e. actors representing other systems or devices. The system is able to respond to such an event.	*Payment system does not respond. The system can propose to use a different method of payment.*	65
Lack of confirmation	A user does not confirm his/her action.	*Administrator does not confirm removing the post.*	12
Missing data	Events concerning missing input data.	*Candidate did not provide personal data*	481
No data defined	System does not have requested data.	*There are no majors available*	259
Payment events	Events identified in a use case "Assign an application fee to a major." They relate either to credit card payment problems (expired card or insufficient resources) or choice of an alternative payment method.	*Credit card has expired.*	18
Redo forbidden	Actor would like to redo some operation that can be performed only once (e.g., registering for the second time).	*Candidate tries to register for the second time.*	103
Wrong data or incorrect data	These are events related to the wrong (in the sense of content) or incorrect (in the sense of syntactic format) input data.	*Provided data is incorrect.*	502

students were familiar with the concept of use cases. They had authored use cases for previous courses in the SE programme curriculum, as well as in the projects they had participated in.

The participants were randomly assigned to two groups: group *G1* was asked to use the *ad hoc* method and group *G2* to use the H4U method.

Object—Software Requirements Specification The object of the experiment was a use-case-based software requirements specification (SRS). To mitigate the risk of

using a specific requirements specification (i.e., too easy, or too difficult to analyze), it was decided to use the benchmark requirements specification[2] (Alchimowicz et al. 2011, 2010). This is an instance of a typical use-case-based SRS, which was derived from the analysis of 524 use cases coming from 16 software development projects.

For the purpose of the experiment the original benchmark specification was purged by removing all the events (only bare scenarios remained). This consisted solely of the use-case headers and main scenarios.

4.2.2 Operation of the Experiment

Prepared Instrumentation The following instrumentation was provided for the participants of the experiment:

- A presentation containing around 30 slides describing the basics of use cases. The *G1* group was presented with the information about the *ad hoc* approach to events identification. The information about the H4U method was presented only to the *G2* group. At the end of the presentation, both groups were informed about the details of the experiment.
- A benchmark specification being the object of the study.
- An editable form where the participants filled in the identified events.
- A questionnaire, prepared in order to identify possible factors which might influence the results of the experiment.

All the materials were provided in English, as all the classes of the Software Engineering programme are conducted in English.

Execution The procedure of the experiment consisted of the following steps:

1. The supervisor of the experiment gave the Presentation (15 min).
2. The participants read the Benchmark Specification and noted all the identified events.
3. After every 10 min the participants were asked to save the file with the events under a different name, and to change the color of the font they were using. This allowed the experimenters to analyze how the set of identified events was changing over in time. The students were given 60 min to finish their task.
4. After the time was finished, the students were asked to fill out the questionnaire.

Data Validation After collection of the data, all questionnaires were reviewed. None were rejected due to incompleteness or errors.

4.2.3 Analysis and Interpretation

Descriptive Statistics Before experimenters proceeded to further analysis, the experimental data was investigated to find and handle outlying observations. After analyzing the descriptive statistics presented in Table 3, we did not observe any potentially outlying observations.

[2]The benchmark requirements specification is available on-line at http://www.ucdb.cs.put.poznan.pl.

Table 3 Descriptive statistics (groups G1 and G2)

	Speed [steps/min]		Accuracy	
	G1	G2	G1	G2
#1	2.63	0.35	0.25	0.36
#2	1.83	0.12	0.22	0.44
#3	2.58	0.40	0.18	0.26
#4	2.65	0.50	0.09	0.22
#5	1.40	0.45	0.13	0.26
#6	1.95	0.33	0.09	0.34
#7	2.62	1.35	0.25	0.31
#8	2.60	1.17	0.14	0.05
#9	1.85	1.07	0.19	0.12
N	9	9	9	9
Min	1.40	0.12	0.09	0.05
1st Qu.	1.85	0.35	0.13	0.22
Median	2.58	0.45	0.18	0.26
Mean	2.23	0.64	0.17	0.26
3rd Qu.	2.62	1.07	0.22	0.34
Max	2.65	1.35	0.25	0.44

The next step in the analysis was the investigation of the samples distributions in order to choose proper statistical hypothesis tests.

The initial observation, based on an analysis of the Q–Q plots, was that the assumption about samples normality might be violated for the speed variable. This suspicion was further confirmed by the Shapiro–Wilk test.[3] Therefore, we decided to use non-parametric statistical tests for speed variable and parametric tests for the accuracy variable.

Hypotheses Testing In order to compare the review speed of the H4U method to the typical *ad hoc* approach, the following hypotheses were formulated. The null hypothesis stated that there is no difference regarding the review speed of the H4U and *ad hoc* methods:

$$H_{a0} : \theta_{Speed(G1)} = \theta_{Speed(G2)} \tag{3}$$

The alternative hypothesis stated that the median review speed differs between these two methods:

$$H_{a1} : \theta_{Speed(G1)} \neq \theta_{Speed(G2)} \tag{4}$$

If accuracy is considered, it is important to investigate whether H4U improves the accuracy of events identification in use cases. Therefore, the following hypotheses were defined:

$$H_{b0} : \mu_{Accuracy(G1)} = \mu_{Accuracy(G2)} \tag{5}$$

[3]The results of the Shapiro–Wilk test ($\alpha = 0.05$)—Speed G1: $W = 0.804$, p-value $= 0.023$ (reject H_0), G2: $W = 0.860$, p-value $= 0.096$ (not reject H_0); Accuracy G1: $W = 0.930$, p-value $= 0.477$ (not reject H_0), G2: $W = 0.974$, p-value $= 0.925$ (not reject H_0).

$$H_{b1} : \mu_{Accuracy(G1)} < \mu_{Accuracy(G2)} \tag{6}$$

To test the hypotheses the Wilcoxon rank-sum test was used to test hypotheses regarding speed and the t-test to test hypotheses concerning accuracy; both with the significance level α set to 0.05.

As the result of the testing procedure, both null hypotheses were rejected (p-values were equal to 4.114e-05 and 3.214e-02).

Power Analysis The observed normalized effect size[4] expressed as Cohen's d coefficient (Cohen 1992) was "large"[5] (Cohen's d for review speed was equal to 3.49; and for accuracy it was equal to 0.96).

Sensitivity analysis, which computes the required effect size with the assumed significance level α, statistical power (1-β), and sample, revealed that in order to achieve a high power for the performed tests (0.95) the effect size should be equal to at least 1.62 (one-tailed t-test) and 1.97 (two-tailed Wilcoxon rank-sum test).

The observed effect size for speed was visibly greater than 1.97, while the one observed for accuracy was lower than 1.62 (the calculated post hoc power was equal to 0.62).

Interpretation Based on the results of the experiment, one could conclude that the H4U method can help to achieve higher accuracy of identification of events in use-case scenarios when used by junior software engineers.

Unfortunately, the higher accuracy of the H4U method was achieved by visibly lowering the speed of the review process (the cumulative number of steps analyzed by both groups in time are presented in Fig. 9).

4.2.4 Threats to Validity

The following threats to validity need to be taken into account when interpreting the results of the experiment.

First, it needs to be noted that only 18 (9 for each group) participants took part in the experiment, which constitutes a small sample.

Second, the participants were students; hence, it might be impossible to generalize the results to the group of professional analysts. However, most of the participants had some practical experience in software houses, so the results might be generalized to junior developers. It is also noteworthy that the participants who used the H4U method were not familiar with this method prior to the experiment; hence, they were learning this approach during the experiment. This factor could possibly have a negative influence on the review speed of the group using the H4U method.

An additional threat to conclusion validity is posed by the disproportion between the number of steps analyzed by participants using different methods. The higher

[4]Please note that the retrospectively calculated effect size only approximates the real effect size in the populations from which the samples were drawn. More information can be found in Zumbo and Hubley (1998).

[5]According to Cohen (1992) effect size is perceived as "small" if the value of d is equal to 0.2, as "medium" if the value of d is equal to 0.5, and as "large" if d is equal to 0.8.

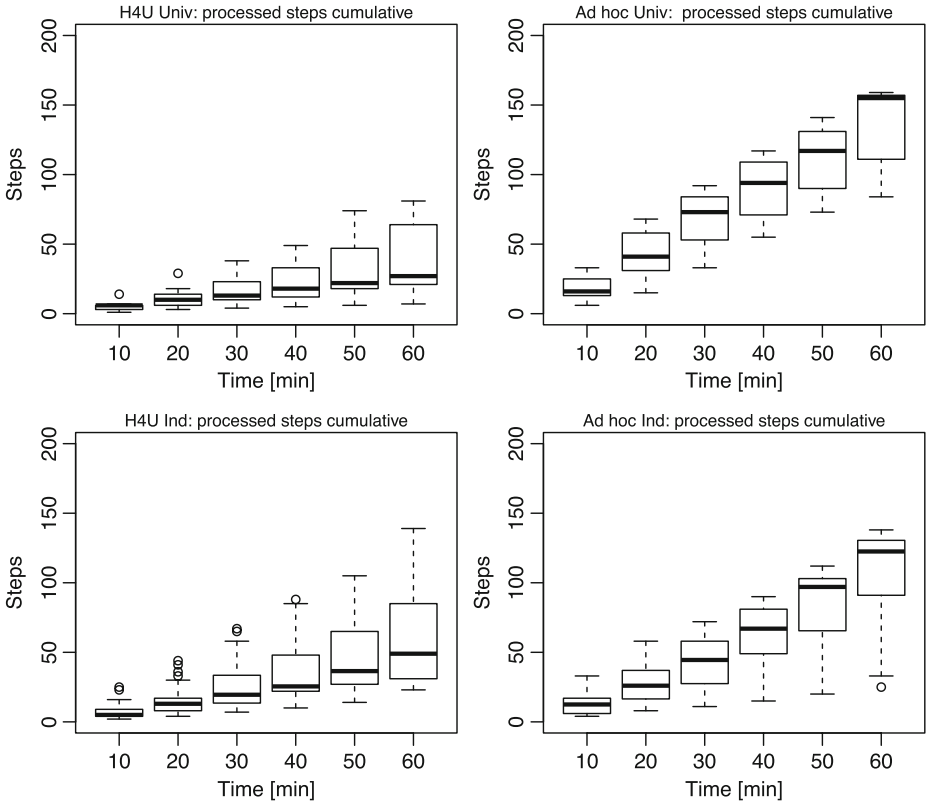

Fig. 9 The cumulative number of steps processed by reviewers in time (Univ—Experiment 1; Ind—Experiment 2)

speed of ad hoc groups enabled them to analyze a greater number of steps than participants using H4U. As a result the number of participants using H4U and contributing proposals of events was decreasing in consecutive steps. As presented in Fig. 10, this could led to an increase in the accuracy of ad hoc groups.

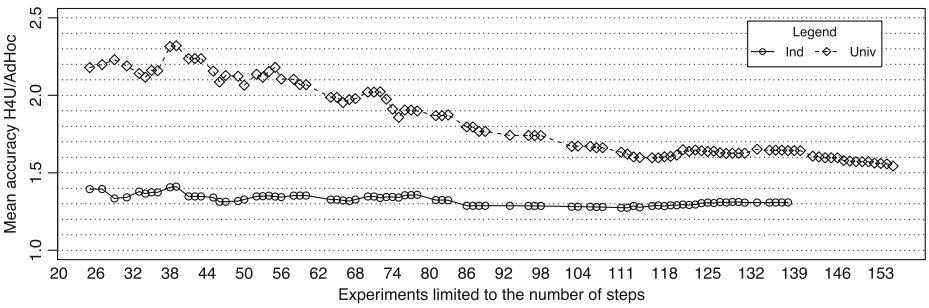

Fig. 10 The ratio between the mean accuracy of H4U and Ad hoc if the scope of experiments was limited to a given number of steps

4.3 Experiment 2: Experienced Reviewers

4.3.1 Experiment Design

Participants The participants in the second experiment were 82 professionals, with experience ranging from 1 to 20 years (mean 6 years, standard deviation 4 years). The group was heterogeneous with respect to roles performed by the participants in projects (software developers & designers, architects, analysts, and project managers). Most of the participants had experience with use cases: 32 % had used them in many projects, 24 % had used them in at least one project, 22 % mentioned only a couple of use cases. There were also participants who did not have any experience related to use-case-based requirements: 14 % had heard about use cases but had not used them, 8 % had never heard about use cases.

The participants were randomly assigned to two groups: group *G3* that was asked to use the *ad hoc* method and group *G4* used the H4U method.

4.3.2 Operation of the Experiment

Prepared Instrumentation The materials were translated into Polish, because we could not assume that all participants are fluent in English. In addition, an additional short survey was prepared to collect demographic information as well as to obtain opinions about the H4U method from the professionals.

Data Validation After collection of the data, all questionnaires were reviewed. We decided to reject 18 observations because of their incompleteness. Unfortunately, some of the participants were interrupted by their duties and had to leave before the end of the experiment. Some decided to identify events and write alternative scenarios to practice the whole use-case development process, which resulted in less time spent on the experiment task of solely identifying events.

4.3.3 Analysis and Interpretation

Descriptive Statistics The experiment data was investigated to find and handle outlying observations. After analyzing the descriptive statistics presented in Table 4, we found 1 potentially outlying observation in group G3 with respect to review speed. However, after deeper investigation of the work, we did not find any justification for rejecting it (the *ad hoc* process does not impose any specific approach to events identification, so the review speed can differ visibly).

In the next step of the analysis samples distributions were investigated in order to choose proper statistical hypothesis tests.

The observation made based on the analysis of the Q–Q plots was that the assumption about samples normality might have been violated for the speed variable. This suspicion was confirmed by the Shapiro–Wilk test.[6] Therefore, again we decided to use the same non-parametric statistical test to test the hypotheses regarding speed and the same parametric statistical test to test the hypotheses concerning accuracy.

[6]The results of the Shapiro–Wilk test ($\alpha = 0.05$)—Speed G3: $W = 0.742$, p-value = 3.953e-06 (reject H_0), G4: $W = 0.868$, p-value = 1.056e-03 (reject H_0); Accuracy G3: $W = 0.973$, p-value = 0.577 (reject H_0), G4: $W = 0.951$, p-value = 0.153 (not reject H_0).

Table 4 Descriptive statistics (groups G3 and G4)

	Speed [steps/min]		Accuracy	
	G3	G4	G3	G4
#1	2.18	2.32	0.19	0.13
#2	2.07	0.85	0.04	0.36
#3	2.20	0.38	0.13	0.45
#4	2.12	2.02	0.13	0.05
#5	0.68	1.97	0.14	0.03
#6	2.17	0.78	0.16	0.23
#7	1.90	0.68	0.11	0.09
#8	0.87	0.38	0.21	0.08
#9	2.23	1.07	0.09	0.10
#10	2.30	1.35	0.24	0.28
#11	2.07	2.07	0.08	0.11
#12	0.68	0.48	0.22	0.23
#13	2.22	1.07	0.09	0.07
#14	2.07	0.48	0.15	0.33
#15	2.02	0.97	0.25	0.19
#16	2.20	0.38	0.28	0.14
#17	2.18	0.42	0.07	0.23
#18	2.02	0.48	0.14	0.09
#19	2.17	0.68	0.22	0.35
#20	1.90	0.55	0.15	0.18
#21	1.07	0.78	0.03	0.26
#22	2.02	0.78	0.07	0.17
#23	2.18	0.78	0.21	0.36
#24	1.97	1.90	0.18	0.14
#25	1.60	1.48	0.12	0.08
#26	1.43	1.83	0.11	0.07
#27	2.02	1.17	0.05	0.31
#28	0.42	0.48	0.15	0.26
#29	0.55	1.07	0.20	0.14
#30	0.87	0.68	0.26	0.15
#31	2.08	2.22	0.11	0.23
#32	2.17	0.85	0.12	0.30
N	32	32	32	32
Min	0.42	0.38	0.03	0.03
1st Qu.	1.56	0.53	0.10	0.10
Median	2.05	0.82	0.14	0.17
Mean	1.77	1.04	0.15	0.19
3rd Qu.	2.17	1.38	0.20	0.26
Max	2.30	2.32	0.28	0.45

Hypotheses Testing In order to compare the review speed and accuracy of the H4U method with the *ad hoc* approach, a set of hypotheses similar to those formulated in Experiment 1 was defined.

The null hypothesis related to review speed stated that there is no difference regarding the review speed of the H4U and *ad hoc* methods:

$$H_{c0} : \theta_{Speed(G3)} = \theta_{Speed(G4)} \tag{7}$$

The alternative hypothesis stated that the median review speed differs between these two methods:

$$H_{c1} : \theta_{Speed(G3)} \neq \theta_{Speed(G4)} \tag{8}$$

In addition, the following hypotheses regarding accuracy were defined:

$$H_{d0} : \mu_{Accuracy(G3)} = \mu_{Accuracy(G4)} \tag{9}$$

$$H_{d1} : \mu_{Accuracy(G3)} < \mu_{Accuracy(G4)} \tag{10}$$

As a result of performing the testing procedure, the null hypothesis H_{c0} regarding review speed was rejected (p-value was equal to 5.055e-05). The second null hypothesis H_{d0} concerning accuracy was also rejected (p-value was equal to 2.477e-02).

Power Analysis The observed normalized effect size expressed as Cohen's d coefficient was "large" for review speed (Cohen's d was equal to 1.21) and "medium" for accuracy (Cohen's d was equal to 0.50).

The sensitivity analysis revealed that in order to achieve a high power for the performed tests (0.95), the effect size should be at least equal to 0.83 (one-tailed t-test) and 0.99 (two-tailed Wilcoxon rank-sum test).

The effect size observed for the accuracy was smaller than 0.83. The calculated post-hoc power of the test was equal to 0.63. The effect size observed for the speed was higher than 0.99.

Interpretation Based on the results of the experiment, one could conclude that the H4U method can help to achieve higher accuracy of identification of events in use-case scenario when used by software engineers (not necessarily analysts).

Unfortunately, again the higher accuracy of the H4U method was achieved by lowering the speed of the review process. The on-average review speed was lowered by 1.7–2.5 times in the case of the H4U method than for *ad hoc* (the cumulative number of steps analyzed by both groups in time are presented in Fig. 9).

In both experiments, we have observed that the students outperformed the professionals in terms of accuracy. This observation was unexpected. We suspect that it is caused by the fact that the students were more up to date with the topic of use cases—they were learning about use cases and practicing during the courses.

Another observation made during both experiments is that the accuracy of events identification is generally low if one considers individual reviewers. The observed mean accuracy ranged from 0.15 (for professionals using the *ad hoc* method) to 0.26 (for students using H4U). This means that a single reviewer was "on-average" able to detect only a quarter of all the events. Therefore, there is a need for a better method of supporting identification of events.

When it comes to the review speed of the approaches, a non-structured *ad hoc* approach seems visibly more efficient. However, in the survey conducted after the experiment, more participants using the *ad hoc* approach had the feeling of solving tasks in a hurry (65 %) than did users of the H4U method (56 %).

Although the H4U method forces users to follow a specific procedure, most of the participants did not perceive it as difficult (∼68 % "definitely no" and "no"; ∼19 % "difficult to say"; ∼13 % "definitely yes" and "yes").

In addition, more than half of the participants stated that they would like to use the method in their next projects (∼53 % "definitely yes" and "yes"; ∼34 % "difficult to say"; ∼13 % "definitely no" and "no")

4.3.4 Threats to Validity

Again, there are some threats to the validity of the study that need to be taken into consideration.

First, there is a threat related to experimental mortality. Among 82 participants 18 did not complete the experiment task, because they were disturbed by their professional duties.

The threat to conclusion validity regarding the disproportion between the number of steps analyzed by both groups seems less serious than in Experiment 1 (see Fig. 10).

There is also a threat related to external validity. Although the experiment involved professionals, only some of them were fulfilling the role of analyst in software development projects.

It also needs to be noted again that the participants who used the H4U method were not familiar with this method prior to the experiment; hence, they were learning this approach during the experiment. This factor could possibly have a negative influence on the review speed of the group using the H4U method.

5 Usage of HAZOP Keywords

The proposed H4U method includes all of the 11 secondary keywords as suggested by Redmill et al. (1999). However, the analysis of the results of the conducted experiments leads to the conclusion that not all of the keywords are used equally. The identified events in both experiments were divided into three sets. The first set contained events which could be detected by the system (according to the *Detectable Conditions* pattern— Adolph et al. 2002a) and which were proper use-case events, the second set consisted of events which could be detected by the system, but were not correct use-case events (e.g. they were non-functional requirements), while the third set contained events which could not be detected by system. The first set is the most important one, as it contains the events which should be included in the use-case based requirements specification. Therefore, the keywords which helped identify the events from the first set might be considered more useful. During the experiments, the participants who used the H4U method were asked to choose which keyword helped them to identify the event (according to the H4U report table as presented in Fig. 5). We counted how many events were identified with usage of a given keyword. As presented in Table 5, some of the keywords were used very often (e.g., NO—about 48 %) while other keywords were used very rarely (e.g., AS WELL AS—less than 1 %) and their contribution to the identification of detectable events was very small. The presence of these very rarely used keywords might have made the analysis more tedious and could negatively influence the results. It appears that some of the keywords might be merged, e.g. MORE and AS WELL AS; some of the participants were using these keywords interchangeably. The pairs EARLY—BEFORE and LATE—AFTER were similar. Although the meaning of the keywords was different they led to the discovery of very similar events. Merging these pairs would decrease the number of keywords, which could shorten the time

Table 5 Percentage usage of HAZOP secondary keywords in both experiments (used in order to identify: detectable use-case events, detectable non-use-case events and undetectable events)

	Detectable use-case events	Detectable non-use-case events	Undetectable events
NO	47.5	44.4	38.5
MORE	5.2	4.2	10.0
LESS	12.1	3.0	12.0
AS WELL AS	0.4	0.8	1.7
PART OF	7.7	3.4	10.7
REVERSE	6.0	11.0	3.8
OTHER THAN	5.8	5.3	10.7
EARLY	2.7	4.2	2.3
LATE	1.9	7.2	4.0
BEFORE	8.5	7.8	4.6
AFTER	2.0	9.1	1.8

needed for events identification with the H4U method. Moreover, as noted in Section 4, the median review speed of the H4U method was not higher than 0.82, so there is a lot of room for improvement.

On the other hand, the introduction of additional, more specific keywords might help to increase the number of identified events. However, it should be noted that the use of keywords might have been different if requirements specifications from different domains were used, hence, for every business domain the presence of particular keywords should be carefully analysed. In case of a business domain which has never been analysed with the H4U method and there is no experience in using the keywords, use the standard set of keywords is advised.

Furthermore, it is possible that the order of the keywords (they were presented in the following order: NO, MORE, LESS, AS WELL AS, PART OF, REVERSE, OTHER THAN, EARLY, LATE, AFTER, BEFORE) might influence how they were used by the participants. There is a need for further research to see how the set of the secondary keywords could be altered to increase the accuracy and review speed of the H4U method.

6 Related Work

Many authors have noted the problem of requirements specification completeness with the focus on events associated with use cases. Wiegers (2003a) included the following question in his inspection checklist for use cases: *Are all known exception conditions documented?*. Mar (1994) identified 5 characteristics concerning requirements specification completeness; one of them reads: *all scenarios and states are recognized*. Cox et al. (2004) suggested 3 types of requirement defects, from the least severe, which have *minimal impact*, to the most severe. To deal with the last type of defects, Cox et al. included the following question in their use-case inspection guidelines: *Alternatives and exceptions should make sense and should be complete. Are they?* In order to solve the issue of incomplete exceptions Carson proposed a 3-step approach for requirements elicitation (Carson 1998). The goal of the third step in this approach is to *verify requirements completeness*, based on the idea of a complementary set of conditions under which the requirements are to be performed (Carson 1995).

Although the processes and approaches mentioned above, like many others (Anda and Sjøberg 2002; Biffl and Halling 2003), seem to emphasize the importance of event-completeness, they do not propose any specific method or technique aimed at identifying events. Our proposal is to adopt the HAZOP approach for this purpose.

HAZOP has already been used in many software development contexts. Redmill et al. (1999) described how HAZOP could be used for the analysis of safety issues in software systems. Jarzębowicz (2007) proposed UML-HAZOP, a method for discovering anomalies in UML diagrams. He is using HAZOP to obtain a checklist against which the quality of UML diagrams can be assessed.

None of the above authors has applied HAZOP to use cases. Allenby and Kelly (2001) were first to apply HAZOP to scenarios and use cases. Their aim was to identify hazards in safety critical systems. In their approach they analyzed pre-, post- and guard-conditions together with system responses. The output of this approach is the possible hazards and failures of the system. Another example of applying HAZOP for analysis of use cases is the work by Srivatanakul et al. (2004). She and her coworkers were interested in identification of security requirements. Their approach requires complete use cases (pre-conditions, main scenarios and extensions, including events and alternative scenarios) as an input. With the help of HAZOP keywords, they determined possible security requirements and policies.

The two above approaches might seem similar to the proposed H4U method as they are applying HAZOP to a use case. In fact they are different. Srivatanakul et al. (2004), and Allenby and Kelly (2001) assume more or less complete use cases as an input, and the output is a report on possible security and safety issues. Moreover, there is a difference in the keywords used by Srivatanakul et al. (2004), and Allenby and Kelly (2001), and H4U. As described in Section 2, H4U is part of a functional requirements elicitation, whereas the approaches presented by Srivatanakul et al. (2004), and Allenby and Kelly (2001) are concerned with nonfunctional requirements. Analysis of use cases has also been studied by others (Alexander 2003; Stålhane and Sindre 2007; Stålhane et al. 2010); however, they are not using HAZOP at all.

7 Conclusions

Event completeness of use cases is one of the quality factors of use-case based requirements specification. Missing events might lead to higher project costs (Stark et al. 1999). Therefore, a method called H4U (HAZOP for Use Cases) aiming at events identification, was proposed in the paper and evaluated experimentally. The method was compared to the *ad hoc* review approach. The main contribution of this paper is that the results of accuracy of events identification in both cases were unsatisfactory, hovewer, usage of the HAZOP-based method lead to the achievement of significantly better accuracy.

The results led to the following more detailed conclusions:

– The maximum average accuracy of events identification in both conducted experiments was 0.26. This shows that events identification is not an easy task,

and many events might be omitted during the analysis, hence new methods and tools are needed to mitigate this problem.
- The proposed HAZOP-based method can help in achieving higher accuracy of events identification. This is true in two distinct environments: IT professionals (0.19 accuracy with H4U and 0.15 with the *at hoc* approach) and students (0.26 accuracy with H4U and 0.17 with the *at hoc* approach).
- H4U enables higher accuracy of events identification, however, it requires more time to perform a review. In both groups, IT professionals and students, the *ad hoc* approach was more efficient in terms of the number of steps analyzed. On average the *ad hoc* approach review speed varied from 1.77 steps analyzed per minute by professionals to 2.23 steps analyzed per minute by junior analysts, while the review speed of the H4U method was respectively 1.04 and 0.64 steps analyzed per minute.
- Students versus professionals. There is a common doubt as to whether an experiment with the participation of students could provide results that remain valid for professionals. One of the most debatable questions regards the differences between the effectiveness of (typically inexperienced) students in comparison to more experienced professionals. Some of the research performed so far indicates that for less complex tasks the difference between students and professionals is minor (Höst et al. 2000). In the experiments described in the paper we observed that master-level SE students can even outperform professionals in some cases.

Acknowledgements We would like to thank all participants of the experiments for their active participation. We express special thanks to Tomasz Bialy, Maciej Dorsz, Andrzej Kurc, Krzysztof Kurowski, Grzegorz Leopold, Piotr Piesik and Maciej Stroiński for their exceptional help.

References

Adolph S, Bramble P, Cockburn A, Pols A (2002a) Patterns for effective use cases. Addison-Wesley, Reading, MA

Adolph S, Bramble P, Cockburn A, Pols A (2002b) Patterns for effective use cases (Agile Software Development). Addison Wesley, Reading, MA

Alchimowicz B, Jurkiewicz J, Nawrocki J, Ochodek M (2010) Building benchmarks for use cases. CAI 29(1):27–44

Alchimowicz B, Jurkiewicz J, Nawrocki J, Ochodek M (2011) Towards use-cases benchmark. In: Software engineering techniques. Lecture notest in computer science, vol 4980. Springer-Verlag, Heidelberg, pp 20–33

Alexander I (2003) Misuse cases: use cases with hostile intent. IEEE Softw 20(1):58–66

Allenby K, Kelly T (2001) Deriving safety requirements using scenarios. In: The 5th IEEE international symposium on Requirements Engineering (RE'01). Society Press, pp 228–235

Anda B, Sjøberg DIK (2002) Towards an inspection technique for use case models. In: Proceedings of the 14th international conference on software engineering and knowledge engineering, pp 127–134

Biffl S, Halling M (2003) Investigating the defect detection effectiveness and cost benefit of nominal inspection teams. IEEE Trans Softw Eng 29(5):385–397

Carson RS (1995) A set theory model for anomaly handling in system requirements analysis. In: Proceedings of NCOSE
Carson RS (1998) Requirements completeness: a deterministic approach. In: Proceedings of 8th annual international INCOSE symposium
Chemical Industries Association (1992) A guide to hazard and operability studies. Published by Chemical Industry Safety and Health Council of the Chemical Industries Association
Cockburn A (2000) Writing effective use cases. Addison-Wesley Professional, Reading, MA
Cohen J (1992) Statistical power analysis. Curr Dir Psychol Sci 1(3):98–101
Cox K, Aurum A, Jeffery R (2004) An experiment in inspecting the quality of use case descriptions. J Res Pract Inf Technol 36(4):211–229
Höst M, Regnell B, Wohlin C (2000) Using students as subject—a comparative study of students and professionals in lead-time impact assessment. Empir Software Eng 5(3):201–214
IEEE (1998) IEEE Std 830-1998, IEEE recommended practice for software requirements specifications
ISO (2011) ISO 25010:1011, Systems and software engineering–systems and software quality requirements and evaluation. ISO Press
Jacobson I (1985) Concepts for modeling large real time systems. Royal Institute of Technology, Dept. of Telecommunication Systems-Computer Systems
Jarzębowicz A (2007) A method for anomaly detection in selected models of information systems. PhD thesis, Gdansk University of Technology
Kruchten P (2003) The rational unified process: an introduction, 3rd edn. Addison-Wesley Professional, Reading, MA
Lawley H (1974) Operability studies and hazard analysis. Chem Eng Prog 70(4):45
Leveson N (1995) Safeware: system safety and computers. Addison-Wesley, Reading, MA
Mar BW (1994) Requirements for development of software requirements. In: Proceedings of NCOSE
Martin J (1983) Managing the data-base environment. The James Martin books on computer systems and telecommunications. Prentice-Hall, Englewood Cliffs, NJ
Nawrocki J, Olek L, Jasinski M, Paliswiat B, Walter B, Pietrzak B, Godek P (2006) Balancing agility and discipline with xprince. In: Rapid integration of software engineering techniques. Lecture notes in computer science, vol 3943. Springer, Berlin/Heidelberg, pp 266–277
Neill C, Laplante P (2003) Requirements engineering: the state of the practice. IEEE Softw 20(6): 40–45
Pumfrey DJ (1999) The principled design of computer system safety analyses. PhD Thesis, University of York
Redmill F, Chudleigh M, Catmur J (1999) System safety: HAZOP and software HAZOP. Wiley, New York
Schlechter W (1995) Process risk assessment—using science to “do it right”. Int J Press Vessel Pip 61(2):479–494
Srivatanakul T, Clark JA, Polack F (2004) Effective security requirements analysis: Hazop and use cases. In: Information security: 7th international conference. Lecture notest in computer science, vol 3225. Springer, Heidelberg, pp 416–427
Stålhane T, Sindre G (2007) A comparison of two approaches to safety analysis based on use cases. In: Conceptual modeling—ER 2007. Lecture notes in computer science, vol 4801. Springer, Berlin/Heidelberg, pp 423–437
Stålhane T, Sindre G, du Bousquet L (2010) Comparing safety analysis based on sequence diagrams and textual use cases. In: Advanced information systems engineering. Lecture notes in computer science, vol 6051. Springer, Heidelberg, pp 165–179
Stark GE, Oman PW, Skillicorn A, Ameele A (1999) An examination of the effects of requirements changes on software maintenance releases. J Softw Maint-Res Pr 11(5):293–309
Wiegers KE (2003a) Inspection checklist for use case documents. In: Proceedings of 9th IEEE international software metrics symposium. IEEE
Wiegers KE (2003b) Software requirement. Microsoft Press
Zumbo B, Hubley A (1998) A note on misconceptions concerning prospective and retrospective power. J Roy Stat Soc D-Sta 47(2):385–388

Jakub Jurkiewicz is a PhD student at Poznan University of Technology. His main research interests lie in the fields of requirements engineering, software quality and agile methodologies. Outside of the university he works as an agile coach, requirements consultant and trainer.

Jerzy Nawrocki received his M.Sc. degree (1980), Ph.D. degree (1984), and Dr. hab. degree (1994)—all in informatics and all from the Poznan University of Technology (PUT), Poznan, Poland. His research interests concern Software Engineering and Project Management. Currently he is the Dean of the Faculty of Computing at PUT, a Vice-Chair of IFIP Technical Committee 2: Software Theory and Practice (since 2011), and an IFIP Board Member (term 2013–2016).

Mirosław Ochodek holds Ph.D. degree and works as an assistant professor in the Institute of Computing Science at the Poznan University of Technology. His main research interests lie in the fields of requirements engineering, software metrics, functional size measurement, and software effort estimation.

Tomasz Głowacki is a teaching assitant at Poznan University of Technology. He holds PhD in computer science. He has over 7 years of experience in area of systems and business analysis, systems technical design, data base design, systems integration and BI systems. Strong algorithmic. He also completed executive education course at Haas School of Business (UC Berkeley).